```
EEEEEEEEEEEEEEE   RRRRRRRRRRR      FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE   RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE   RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEE               RRR    RRR        FFF
EEE               RRR    RRR        FFF
EEE               RRR    RRR        FFF
EEE               RRR       RRR     FFF
EEE               RRR       RRR     FFF
EEE               RRR       RRR     FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
```

FILEID**MFTAPE

```
MM      MM  FFFFFFFFFF  TTTTTTTTTT    AAAAAA    PPPPPPPP    EEEEEEEEEE
MM      MM  FFFFFFFFFF  TTTTTTTTTT    AAAAAA    PPPPPPPP    EEEEEEEEEE
MMMM  MMMM  FF              TT       AA    AA   PP      PP  EE
MMMM  MMMM  FF              TT       AA    AA   PP      PP  EE
MM  MM  MM  FF              TT       AA    AA   PP      PP  EE
MM  MM  MM  FF              TT       AA    AA   PP      PP  EE
MM      MM  FFFFFFFF        TT       AA    AA   PPPPPPPP    EEEEEEEE
MM      MM  FFFFFFFF        TT       AA    AA   PPPPPPPP    EEEEEEEE
MM      MM  FF              TT       AAAAAAAAAA   PP        EE
MM      MM  FF              TT       AAAAAAAAAA   PP        EE
MM      MM  FF              TT       AA    AA   PP        EE              ....
MM      MM  FF              TT       AA    AA   PP        EE              ....
MM      MM  FF              TT       AA    AA   PP        EEEEEEEEEE      ....
MM      MM  FF              TT       AA    AA   PP        EEEEEEEEEE      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

MFT

```
084
084
084
084
084
084
084
085
085
085
085
085
085
085
085
085
086
086
086
086
086
086
086
086
086
086
086
087
087
087
087
087
087
087
087
088
088
088
088
088
088
088
088
088
088
089
089
089
089
089
089
089
089
089
089
089
```

```
0001          subroutine mftape (lun)
0002    C
0003    C Version:      'V04-000'
0004    C
0005    C*****************************************************************************
0006    C*                                                                           *
0007    C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0008    C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0009    C*  ALL RIGHTS RESERVED.                                                     *
0010    C*                                                                           *
0011    C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0012    C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0013    C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0014    C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0015    C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0016    C*  TRANSFERRED.                                                             *
0017    C*                                                                           *
0018    C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0019    C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0020    C*  CORPORATION.                                                             *
0021    C*                                                                           *
0022    C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0023    C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0024    C*                                                                           *
0025    C*                                                                           *
0026    C*****************************************************************************
0027    C
0028
0029    C
0030    C       Functional description:
0031    C
0032    C       This module displays error log entries for the TU78 tape drive.
0033    C
0034    C       Modified by:
0035    C
0036    C       V03-004 SAR0226         Sharon A. Reynolds,     28-Mar-1984
0037    C               Changed the call to UCB$L_OWNUIC to ORB$L_OWNER.
0038    C
0039    C       V03-003 SAR0085         Sharon A. Reynolds,     20-Jun-1983
0040    C               Changed the carriage control in the 'format' statements
0041    C               for use with ERF.
0042    C
0043    C       V03-002 SAR0044         Sharon A. Reynolds,     9-Jun-1983
0044    C               Remove brief and cryptic support.
0045    C
0046    C       v03-001 BP0005          Brian Porter,           01-JUN-1982
0047    C               Corrected attention interrupts.
0048    C
0049    C       v02-004 BP0004          Brian Porter,           07-FEB-1982
0050    C               Corrected call to mba_control_etc..
0051    C
0052    C       v02-003 BP0003          Brian Porter,           18-NOV-1981
0053    C               Added new mba code.  Minor edit.
0054    C
0055    C       v02-002 BP0002          Brian Porter,           06-NOV-1981
0056    C               Added 'device attention' support.
0057    C
```

```
0058     C       v02-001 BP0001          Brian Porter,           01-JUL-1981
0059     C               Added call to LOGGER and DHEAD.
0060     C**
0061
0062     C
0063     C       This is the format of the error log entry for the TU78.
0064     C
0065     C
0066     C       +-------------------------------+
0067     C       |                               |
0068     C       +--                           --+
0069     C       |                               |
0070     C       +--          header           --+
0071     C       |                               |
0072     C       +--                           --+
0073     C       |                               |
0074     C       +-------------------------------+
0075     C       |            mf cs1             |
0076     C       +-------------------------------+
0077     C       |            mf is              |
0078     C       +-------------------------------+
0079     C       |            mf tc              |
0080     C       +-------------------------------+
0081     C       |            mf mr1             |
0082     C       +-------------------------------+
0083     C       |            mf ab              |
0084     C       +-------------------------------+
0085     C       |            mf bc              |
0086     C       +-------------------------------+
0087     C       |            mf dt              |
0088     C       +-------------------------------+
0089     C       |            mf ds              |
0090     C       +-------------------------------+
0091     C       |            mf sn              |
0092     C       +-------------------------------+
0093     C       |            mf mr2             |
0094     C       +-------------------------------+
0095     C       |            mf mr3             |
0096     C       +-------------------------------+
0097     C       |            mf ndta            |
0098     C       +-------------------------------+
0099     C       |            mf ndt0            |
0100     C       +-------------------------------+
0101     C       |            mf ndt1            |
0102     C       +-------------------------------+
0103     C       |            mf ndt2            |
0104     C       +-------------------------------+
0105     C       |            mf ndt3            |
0106     C       +-------------------------------+
0107     C       |            mf id              |
0108     C       +-------------------------------+
0109     C       |         ucb$l_mf_cmd          |
0110     C       +-------------------------------+
0111     C       |                               |
0112     C       +--                           --+
0113     C       |                               |
0114     C       +--                           --+
```

```
0115    C       /          60 bytes            /
0116    C       /  extended sense information  /
0117    C       +--                          --+
0118    C       |                              |
0119    C       +--                          --+
0120    C       |                              |
0121    C       +------------------------------+
0122    C
0123    C**
0124
0125            include 'src$:msghdr.for /nolist'
0184            include 'src$:deverr.for /nolist'
0285
0286
0287            byte            lun
0288            byte            mf_exsns(60)
0289
0290            integer*4       field
0291            integer*4       dt_fcode
0292            integer*4       ndt_fcode
0293            integer*4       dt_cmdaddr
0294            integer*4       ndt_cmdaddr
0295            integer*4       dt_intcode
0296            integer*4       ndt_intcode
0297            integer*4       dt_function
0298            integer*4       ndt_function
0299            integer*4       attn_bit_this_tm78
0300            integer*4       compress4
0301            integer*4       compressc
0302            integer*4       lib$extzv
0303            integer*4       lib$extv
0304
0305            logical*1       diagnostic_mode
0306
0307            integer*4       ucb$l_mf_cmd
0308            integer*4       ucb_function
0309            integer*4       selected_map_register
0310            integer*4       adapter_registers(7)
0311            integer*4       mf_csl
0312            integer*4       mf_is
0313            integer*4       mf_tc
0314            integer*4       mf_mrl
0315            integer*4       mf_ab
0316            integer*4       mf_bc
0317            integer*4       mf_dt
0318            integer*4       mf_ds
0319            integer*4       mf_sn
0320            integer*4       mf_mr2
0321            integer*4       mf_mr3
0322            integer*4       mf_ndta
0323            integer*4       mf_ndt(0:3)
0324            integer*4       mf_id
0325            integer*4       ucb_unit_number
0326
0327            equivalence     (emb$l_dv_regsav(0),adapter_registers)
0328            equivalence     (emb$l_dv_regsav(7),mf_csl)
0329            equivalence     (emb$l_dv_regsav(8),mf_is)
```

```
0330            equivalence     (emb$l_dv_regsav(9),mf_tc)
0331            equivalence     (emb$l_dv_regsav(10),mf_mr1)
0332            equivalence     (emb$l_dv_regsav(11),mf_ab)
0333            equivalence     (emb$l_dv_regsav(12),mf_bc)
0334            equivalence     (emb$l_dv_regsav(13),mf_dt)
0335            equivalence     (emb$l_dv_regsav(14),mf_ds)
0336            equivalence     (emb$l_dv_regsav(15),mf_sn)
0337            equivalence     (emb$l_dv_regsav(16),mf_mr2)
0338            equivalence     (emb$l_dv_regsav(17),mf_mr3)
0339            equivalence     (emb$l_dv_regsav(18),mf_ndta)
0340            equivalence     (emb$l_dv_regsav(19),mf_ndt(0))
0341            equivalence     (emb$l_dv_regsav(23),mf_id)
0342            equivalence     (emb$l_dv_regsav(24),ucb$l_mf_cmd)
0343            equivalence     (emb$l_dv_regsav(26),mf_exsns)
0344
0345            character*27    v1interrupt_code(13)
0346            DATA    V1INTERRUPT_CODE(1)     /'OPERATION COMPLETED*'/
0347            DATA    V1INTERRUPT_CODE(2)     /'UNEXPECTED TAPE MARK FOUND*'/
0348            DATA    V1INTERRUPT_CODE(3)     /'UNEXPECTED BOT*'/
0349            DATA    V1INTERRUPT_CODE(4)     /'TAPE AT OR BEYOND EOT*'/
0350            DATA    V1INTERRUPT_CODE(5)     /'UNEXPECTED LOGICAL EOT*'/
0351            DATA    V1INTERRUPT_CODE(6)     /'NO-OP COMPLETED*'/
0352            DATA    V1INTERRUPT_CODE(7)     /'TAPE UNIT REWINDING*'/
0353            DATA    V1INTERRUPT_CODE(8)     /'TAPE UNIT WRITE PROTECTED*'/
0354            DATA    V1INTERRUPT_CODE(9)     /'SUB-SYSTEM NOT READY*'/
0355            DATA    V1INTERRUPT_CODE(10)    /'TAPE UNIT NOT AVAILABLE*'/
0356            DATA    V1INTERRUPT_CODE(11)    /'TAPE UNIT OFF-LINE*'/
0357            DATA    V1INTERRUPT_CODE(12)    /'TAPE UNIT NON-EXISTENT*'/
0358            DATA    V1INTERRUPT_CODE(13)    /'SUB-SYSTEM NOT CAPABLE*'/
0359
0360            character*31    v2interrupt_code(15:28)
0361            DATA    V2INTERRUPT_CODE(15)    /'TAPE UNIT ON-LINE TRANSITION*'/
0362            DATA    V2INTERRUPT_CODE(16)    /'LONG RECORD*'/
0363            DATA    V2INTERRUPT_CODE(17)    /'SHORT RECORD*'/
0364            DATA    V2INTERRUPT_CODE(18)    /'RETRY*'/
0365            DATA    V2INTERRUPT_CODE(19)    /'RETRY OPPOSITE*'/
0366            DATA    V2INTERRUPT_CODE(20)    /'UNREADABLE*'/
0367            DATA    V2INTERRUPT_CODE(21)    /'ERROR (RETRIES SUPPRESSED)*'/
0368            DATA    V2INTERRUPT_CODE(22)    /'EOT ERROR (RETRIES SUPPRESSED)*'/
0369            DATA    V2INTERRUPT_CODE(23)    /'BAD TAPE*'/
0370            DATA    V2INTERRUPT_CODE(24)    /'TM FAULT A*'/
0371            DATA    V2INTERRUPT_CODE(25)    /'TU FAULT A*'/
0372            DATA    V2INTERRUPT_CODE(26)    /'TM FAULT B*'/
0373            DATA    V2INTERRUPT_CODE(27)    /'TU FAULT B*'/
0374            DATA    V2INTERRUPT_CODE(28)    /'MASSBUS CONTROL BUS FAULT*'/
0375
0376
0377            character*18    fcode_intcode1(0:1)
0378            DATA    FCODE_INTCODE1(0)       /'EXSNS NOT UPDATED*'/
0379            DATA    FCODE_INTCODE1(1)       /'EXSNS UPDATED*'/
0380
0381            character*31    fcode_intcode3(3)
0382            DATA    FCODE_INTCODE3(1)       /'TAPE ALREADY AT BOT*'/
0383            DATA    FCODE_INTCODE3(2)       /'BOT DETECTED AFTER TAPE MOTION*'/
0384            DATA    FCODE_INTCODE3(3)       /'ARA ID DETECTED*'/
0385
0386            character*32    fcode_intcode9(3)
```

```
0387            DATA    FCODE_INTCODE9(1)       /'TAPE UNIT ON-LINE BUT NOT READY*'/
0388            DATA    FCODE_INTCODE9(2)       /'FATAL ERROR, TM CLEAR REQUIRED*'/
0389            DATA    FCODE_INTCODE9(3)       /'ACCESS ONLY, DRIVE BUSY*'/
0390
0391            character*24    fcode_intcode13(4)
0392            DATA    FCODE_INTCODE13(1)      /'BLANK TAPE*'/
0393            DATA    FCODE_INTCODE13(2)      /'ID MARKER NOT PE OR GCR*'/
0394            DATA    FCODE_INTCODE13(3)      /'ARA ID NOT FOUND*'/
0395            DATA    FCODE_INTCODE13(4)      /'NO GAP AFTER ID BURST*'/
0396
0397            character*28    fcode_intcode18(11)
0398            DATA    FCODE_INTCODE18(1)      /'GCR WRITE OPERATION FAILURE*'/
0399            DATA    FCODE_INTCODE18(2)      /'GCR READ OPERATION FAILURE*'/
0400            DATA    FCODE_INTCODE18(3)      /'PE READ OPERATION FAILURE*'/
0401            DATA    FCODE_INTCODE18(4)      /'PE WRITE OPERATION FAILURE*'/
0402            DATA    FCODE_INTCODE18(5)      /'ECCSTA BIT(S) SET*'/
0403            DATA    FCODE_INTCODE18(6)      /'PE WRITE OPERATION FAILURE*'/
0404            DATA    FCODE_INTCODE18(7)      /'GCR WRITE OPERATION FAILURE*'/
0405            DATA    FCODE_INTCODE18(8)      /'RSTAT CONTAINS BAD CODE*'/
0406            DATA    FCODE_INTCODE18(9)      /'PE WRITE OPERATION FAILURE*'/
0407            DATA    FCODE_INTCODE18(10)     /'MASSBUS DATA PARITY ERROR*'/
0408            DATA    FCODE_INTCODE18(11)     /'RETRY OPPOSITE FAILURE*'/
0409
0410            character*29    fcode_intcode28(2)
0411            DATA    FCODE_INTCODE28(1)      /'MASSBUS CONTROL PARITY ERROR*'/
0412            DATA    FCODE_INTCODE28(2)      /'ILLEGAL REGISTER REFERENCE*'/
0413
0414            character*31    v1ndt_function(19)
0415            DATA    V1NDT_FUNCTION(1)       /'NO-OPERATION*'/
0416            DATA    V1NDT_FUNCTION(2)       /'UNLOAD*'/
0417            DATA    V1NDT_FUNCTION(3)       /'REWIND*'/
0418            DATA    V1NDT_FUNCTION(4)       /'SENSE*'/
0419            DATA    V1NDT_FUNCTION(5)       /'DSE*'/
0420            DATA    V1NDT_FUNCTION(6)       /'WRITE TM (PE)*'/
0421            DATA    V1NDT_FUNCTION(7)       /'WRITE TM (GCR)*'/
0422            DATA    V1NDT_FUNCTION(8)       /'SPACE FORWARD RECORD(S)*'/
0423            DATA    V1NDT_FUNCTION(9)       /'SPACE REVERSE RECORD(S)*'/
0424            DATA    V1NDT_FUNCTION(10)      /'SPACE FORWARD FILE(S)*'/
0425            DATA    V1NDT_FUNCTION(11)      /'SPACE REVERSE FILE(S)*'/
0426            DATA    V1NDT_FUNCTION(12)      /'SPACE FORWARD EITHER*'/
0427            DATA    V1NDT_FUNCTION(13)      /'SPACE REVERSE EITHER*'/
0428            DATA    V1NDT_FUNCTION(14)      /'EXTENDED RECORD GAP, SET PE*'/
0429            DATA    V1NDT_FUNCTION(15)      /'EXTENDED RECORD GAP, SET GCR*'/
0430            DATA    V1NDT_FUNCTION(16)      /'CLOSE FILE, PE*'/
0431            DATA    V1NDT_FUNCTION(17)      /'CLOSE FILE, GCR*'/
0432            DATA    V1NDT_FUNCTION(18)      /'SPACE TO LOGICAL EOT*'/
0433            DATA    V1NDT_FUNCTION(19)      /'SPACE FORWARD FILE/LOGICAL EOT*'/
0434
0435            character*20    v1dt_function(20:20)
0436            DATA    V1DT_FUNCTION(20)       /'WRITE CHECK FORWARD*'/
0437
0438            character*20    v2dt_function(23:25)
0439            DATA    V2DT_FUNCTION(23)       /'WRITE CHECK REVERSE*'/
0440            DATA    V2DT_FUNCTION(24)       /'WRITE PE*'/
0441            DATA    V2DT_FUNCTION(25)       /'WRITE GCR*'/
0442
0443            character*15    v3dt_function(28:29)
```

```
0444            DATA    V3DT_FUNCTION(28)       /'READ FORWARD*'/
0445            DATA    V3DT_FUNCTION(29)       /'EXTENDED SENSE*'/
0446
0447            character*13    v4dt_function(31:31)
0448            DATA    V4DT_FUNCTION(31)       /'READ REVERSE*'/
0449
0450            character*23    v1mf_dt(10:11)
0451            DATA    V1MF_DT(10)             /'SLAVE PRESENT*'/
0452            DATA    V1MF_DT(11)             /'DRIVE REQUEST REQUIRED*'/
0453
0454            character*22    v2mf_dt(14:15)
0455            DATA    V2MF_DT(14)             /'TAPE DRIVE*'/
0456            DATA    V2MF_DT(15)             /'NOT BLOCK ADDRESSABLE*'/
0457
0458            character*32    v1mf_ds(4:4)
0459            DATA    V1MF_DS(4)              /'PERFORMING ERASE OF DES COMMAND*'/
0460
0461            character*15    v2mf_ds(6:15)
0462            DATA    V2MF_DS(6)              /'TM SHARED*'/
0463            DATA    V2MF_DS(7)              /'TM AVAILABLE*'/
0464            DATA    V2MF_DS(8)              /'WRITE PROTECTED*'/
0465            DATA    V2MF_DS(9)              /'BEYOND EOT*'/
0466            DATA    V2MF_DS(10)             /'TAPE AT BOT*'/
0467            DATA    V2MF_DS(11)             /'PE MODE*'/
0468            DATA    V2MF_DS(12)             /'REWINDING*'/
0469            DATA    V2MF_DS(13)             /'ON-LINE*'/
0470            DATA    V2MF_DS(14)             /'POWER APPLIED*'/
0471            DATA    V2MF_DS(15)             /'READY*'/
0472
0473            character*27    mf_tc_format(0:6)
0474            DATA    MF_TC_FORMAT(0)         /'11 NORMAL*'/
0475            DATA    MF_TC_FORMAT(1)         /'15 NORMAL*'/
0476            DATA    MF_TC_FORMAT(2)         /'10 COMPATIBLE*'/
0477            DATA    MF_TC_FORMAT(3)         /'10 CORE DUMP*'/
0478            DATA    MF_TC_FORMAT(4)         /'10 HIGH DENSITY COMPATIBLE*'/
0479            DATA    MF_TC_FORMAT(5)         /'IMAGE*'/
0480            DATA    MF_TC_FORMAT(6)         /'10 HIGH DENSITY DUMP*'/
0481
0482            character*29    v1mf_id(8:15)
0483            DATA    V1MF_ID(8)              /'HOLD*'/
0484            DATA    V1MF_ID(9)              /'HLDA*'/
0485            DATA    V1MF_ID(10)             /'EVEN PARITY*'/
0486            DATA    V1MF_ID(11)             /'MASSBUS CONTROL PARITY ERROR*'/
0487            DATA    V1MF_ID(12)             /'ILLEGAL REGISTER REFERENCE*'/
0488            DATA    V1MF_ID(13)             /'MICRO PROC. ROM PARITY ERROR*'/
0489            DATA    V1MF_ID(14)             /'TM CLEAR*'/
0490            DATA    V1MF_ID(15)             /'TM READY*'/
0491
0492            character*14    v1mf_is(8:8)
0493            DATA    V1MF_IS(8)              /'DRIVE PRESENT*'/
0494
0495            character*29    v1mf_tc(15:15)
0496            DATA    V1MF_TC(15)             /'SUPPRESS ERROR REPOSITIONING*'/
0497
0498            character*7     v1mf_cs1(0:0)
0499            DATA    V1MF_CS1(0)             /'GO BIT*'/
0500
```

```
0501            character*16    v2mf_cs1(11:11)
0502            DATA    V2MF_CS1(11)            /'DRIVE AVAILABLE*'/
0503
0504            character*22    v1mf_exsns5(0:7)
0505            DATA    V1MF_EXSNS5(0)          /'WRITE FAIL P*'/
0506            DATA    V1MF_EXSNS5(1)          /'STATISTICS SELECT*'/
0507            DATA    V1MF_EXSNS5(2)          /'CLOCK STOPPED*'/
0508            DATA    V1MF_EXSNS5(3)          /'BEGINNING OF PREAMBLE*'/
0509            DATA    V1MF_EXSNS5(4)          /'DATA NOT READY*'/
0510            DATA    V1MF_EXSNS5(5)          /'PREAMBLE ERROR*'/
0511            DATA    V1MF_EXSNS5(6)          /'STATUS VALID*'/
0512            DATA    V1MF_EXSNS5(7)          /'VELOCITY ok*'/
0513
0514            character*17    v1mf_exsns13(0:7)
0515            DATA    V1MF_EXSNS13(0)         /'AMTIE P*'/
0516            DATA    V1MF_EXSNS13(1)         /'NOT DONE P*'/
0517            DATA    V1MF_EXSNS13(2)         /'ILLEGAL P*'/
0518            DATA    V1MF_EXSNS13(3)         /'MARK 2 P*'/
0519            DATA    V1MF_EXSNS13(4)         /'END P*'/
0520            DATA    V1MF_EXSNS13(5)         /'POST P*'/
0521            DATA    V1MF_EXSNS13(6)         /'DATA P*'/
0522            DATA    V1MF_EXSNS13(7)         /'CORRECTED DATA P*'/
0523
0524            character*30    v1mf_exsns18(0:7)
0525            DATA    V1MF_EXSNS18(0)         /'SINGLE TRACK ERROR CORRECTION*'/
0526            DATA    V1MF_EXSNS18(1)         /'TWO TRACK ERROR CORRECTION*'/
0527            DATA    V1MF_EXSNS18(2)         /'UNCORRECTABLE*'/
0528            DATA    V1MF_EXSNS18(3)         /'POINTER MISMATCH*'/
0529            DATA    V1MF_EXSNS18(4)         /'ACRC ERROR*'/
0530            DATA    V1MF_EXSNS18(5)         /'AMTIE OCCURRED*'/
0531            DATA    V1MF_EXSNS18(6)         /'ECC ROM PARITY ERROR*'/
0532            DATA    V1MF_EXSNS18(7)         /'CRC ERROR*'/
0533
0534            character*21    v1mf_exsns30(0:7)
0535            DATA    V1MF_EXSNS30(0)         /'AMTIE PARITY BIT*'/
0536            DATA    V1MF_EXSNS30(1)         /'READ PARITY BIT*'/
0537            DATA    V1MF_EXSNS30(2)         /'WCS PARITY BIT*'/
0538            DATA    V1MF_EXSNS30(3)         /'TACHOMETER*'/
0539            DATA    V1MF_EXSNS30(4)         /'TAPE UNIT PRESENT*'/
0540            DATA    V1MF_EXSNS30(5)         /'COMMAND PARITY ERROR*'/
0541            DATA    V1MF_EXSNS30(6)         /'WRITE DATA STROBE*'/
0542            DATA    V1MF_EXSNS30(7)         /'STATUS PARITY ERROR*'/
0543
0544            character*18    v1mf_exsns32(3:7)
0545            DATA    V1MF_EXSNS32(3)         /'MASSBUS ATTN*'/
0546            DATA    V1MF_EXSNS32(4)         /'ILLEGAL REGISTER*'/
0547            DATA    V1MF_EXSNS32(5)         /'CAS PARITY ERROR*'/
0548            DATA    V1MF_EXSNS32(6)         /'TM READY*'/
0549            DATA    V1MF_EXSNS32(7)         /'CONTENTION*'/
0550
0551            character*15    v1mf_exsns33(0:7)
0552            DATA    V1MF_EXSNS33(0)         /'ONLINE*'/
0553            DATA    V1MF_EXSNS33(1)         /'-5V OK*'/
0554            DATA    V1MF_EXSNS33(2)         /'LEFT*'/
0555            DATA    V1MF_EXSNS33(3)         /'MASSBUS FAIL*'/
0556            DATA    V1MF_EXSNS33(4)         /'MASSBUS INIT*'/
0557            DATA    V1MF_EXSNS33(5)         /'MASSBUS DEMAND*'/
```

```
0558            DATA    V1MF_EXSNS33(6)         /'MASSBUS TRA*'/
0559            DATA    V1MF_EXSNS33(7)         /'MASSBUS ATTN*'/
0560
0561            character*21    v1mf_exsns34(0:7)
0562            DATA    V1MF_EXSNS34(0)         /'MASSBUS SCLK*'/
0563            DATA    V1MF_EXSNS34(1)         /'SCLK OUT*'/
0564            DATA    V1MF_EXSNS34(2)         /'MASSBUS RUN*'/
0565            DATA    V1MF_EXSNS34(3)         /'MASSBUS EXC*'/
0566            DATA    V1MF_EXSNS34(4)         /'MASSBUS EBL*'/
0567            DATA    V1MF_EXSNS34(5)         /'MASSBUS OCC*'/
0568            DATA    V1MF_EXSNS34(6)         /'MASSBUS WCLK*'/
0569            DATA    V1MF_EXSNS34(7)         /'MASSBUS WRITE ENABLE*'/
0570
0571            character*29    v1mf_exsns47(3:7)
0572            DATA    V1MF_EXSNS47(3)         /'DR READ PARITY ERROR*'/
0573            DATA    V1MF_EXSNS47(4)         /'WMC ROM PARITY ERROR*'/
0574            DATA    V1MF_EXSNS47(5)         /'ERROR*'/
0575            DATA    V1MF_EXSNS47(6)         /'DR MASSBUS DATA PARITY ERROR*'/
0576            DATA    V1MF_EXSNS47(7)         /'DR NO WRITE CLOCK*'/
0577
0578            character*33    v1mf_exsns48(0:7)
0579            DATA    V1MF_EXSNS48(0)         /'TRANSLATOR ROM PARITY ERROR*'/
0580            DATA    V1MF_EXSNS48(1)         /'PE WRITE PARITY ERROR*'/
0581            DATA    V1MF_EXSNS48(2)         /'MASSBUS B LOGIC NOT PRESENT*'/
0582            DATA    V1MF_EXSNS48(3)         /'MASSBUS A LOGIC NOT PRESENT*'/
0583            DATA    V1MF_EXSNS48(4)         /'WRITE DATA REGISTER PARITY BIT*'/
0584            DATA    V1MF_EXSNS48(5)         /'POWER OK*'/
0585            DATA    V1MF_EXSNS48(6)         /'MICRO-COMPUTER ROM PARITY ERROR*'/
0586            DATA    V1MF_EXSNS48(7)         /'MASSBUS PORT SELECT*'/
0587
0588            character*13    v1mf_exsns49(0:7)
0589            DATA    V1MF_EXSNS49(0)         /'FILE PROTECT*'/
0590            DATA    V1MF_EXSNS49(1)         /'EOT*'/
0591            DATA    V1MF_EXSNS49(2)         /'BOT*'/
0592            DATA    V1MF_EXSNS49(3)         /'PES*'/
0593            DATA    V1MF_EXSNS49(4)         /'REWINDING*'/
0594            DATA    V1MF_EXSNS49(5)         /'ONLINE*'/
0595            DATA    V1MF_EXSNS49(6)         /'READY ON*'/
0596            DATA    V1MF_EXSNS49(7)         /'READY*'/
0597
0598            character*14    v1mf_exsns50(0:7)
0599            DATA    V1MF_EXSNS50(0)         /'DSE*'/
0600            DATA    V1MF_EXSNS50(1)         /'MOT*'/
0601            DATA    V1MF_EXSNS50(2)         /'LWR*'/
0602            DATA    V1MF_EXSNS50(3)         /'WRITE INHIBIT*'/
0603            DATA    V1MF_EXSNS50(4)         /'WRITE*'/
0604            DATA    V1MF_EXSNS50(5)         /'REVERSE*'/
0605            DATA    V1MF_EXSNS50(6)         /'FORWARD*'/
0606            DATA    V1MF_EXSNS50(7)         /'MANUAL TEST*'/
0607
0608            character*21    v1mf_exsns51(5:7)
0609            DATA    V1MF_EXSNS51(5)         /'ARA ERROR*'/
0610            DATA    V1MF_EXSNS51(6)         /'PEC*'/
0611            DATA    V1MF_EXSNS51(7)         /'CMD/STA PARITY ERROR*'/
0612
0613            character*20    v1mf_exsns54(2:7)
0614            DATA    V1MF_EXSNS54(2)         /'HIGH READ THRESHOLD*'/
```

```
0615            DATA    V1MF_EXSNS54(3)         /'TACH*'/
0616            DATA    V1MF_EXSNS54(4)         /'EOT DETECTED*'/
0617            DATA    V1MF_EXSNS54(5)         /'READ ENABLE*'/
0618            DATA    V1MF_EXSNS54(6)         /'NOT WRITE*'/
0619            DATA    V1MF_EXSNS54(7)         /'NOT WRITE BIT 4*'/
0620
0621            character*26    v1mf_exsns56(5:7)
0622            DATA    V1MF_EXSNS56(5)         /'INITIAL DIRECTION REVERSE*'/
0623            DATA    V1MF_EXSNS56(6)         /'INITIAL COMMAND READ*'/
0624            DATA    V1MF_EXSNS56(7)         /'LAST RETRY OPPOSITE*'/
0625
0626            character*30    v1mf_exsns57(0:7)
0627            DATA    V1MF_EXSNS57(0)         /'DSE IN PROGRESS*'/
0628            DATA    V1MF_EXSNS57(1)         /'REWIND IN PROGRESS*'/
0629            DATA    V1MF_EXSNS57(2)         /'TAPE PRESENT, POWER ON*'/
0630            DATA    V1MF_EXSNS57(3)         /'NDT FROM mASSBUS IN PROGRESS*'/
0631            DATA    V1MF_EXSNS57(4)         /'LAST DIRECTION REVERSE*'/
0632            DATA    V1MF_EXSNS57(5)         /'LAST OPERATION INCLUDED WRITE*'/
0633            DATA    V1MF_EXSNS57(6)         /'LAST RECORD TAPE MARK*'/
0634            DATA    V1MF_EXSNS57(7)         /'LAST MASSBUS COMMAND PORT B*'/
0635
0636            character*8     v1tu_selx(2:5,0:1)
0637            DATA    V1TU_SELX(2,0)          /'1 WRITE*'/
0638            DATA    V1TU_SELX(3,0)          /'0 WRITE*'/
0639            DATA    V1TU_SELX(4,0)          /'1 READ*'/
0640            DATA    V1TU_SELX(5,0)          /'0 READ*'/
0641
0642            DATA    V1TU_SELX(2,1)          /'3 WRITE*'/
0643            DATA    V1TU_SELX(3,1)          /'2 WRITE*'/
0644            DATA    V1TU_SELX(4,1)          /'3 READ*'/
0645            DATA    V1TU_SELX(5,1)          /'2 READ*'/
0646
0647
0648            call frctof (lun)
0649
0650            call dhead1 (lun,'MASSBUS')
0651
0652            diagnostic_mode = .false.
0653
0654            if (lib$extzv(5,1,mf_ds) .eq. 1) diagnostic_mode = .true.
0655
0656            dt_cmdaddr = lib$extzv (0,2,mf_tc)
0657
0658            dt_function = lib$extzv (1,5,mf_cs1)
0659
0660            ucb_function  = lib$extzv (1,5,ucb$l_mf_cmd)
0661
0662            ucb_unit_number = lib$extzv (0,8,emb$w_dv_unit)
0663
0664            attn_bit_this_tm78 = lib$extzv (lib$extzv(0,8,emb$b_dv_slave),1,mf_ab)
0665
0666            if (dt_cmdaddr .eq. ucb_unit_number
0667           1 .and.
0668           2 dt_function .eq. ucb_function
0669           3 .and.
0670           4 attn_bit_this_tm78 .eq. 0) then
0671
```

```
0672            ndt_function = 0
0673
0674            call mba_control_registers (lun,5,adapter_registers,
0675            1 selected_map_register)
0676
0677            call mba_mapping_register (lun,selected_map_register,
0678            1 adapter_registers(6))
0679
0680            if (selected_map_register .gt. 0) then
0681
0682            call mba_mapping_register (lun,(selected_map_register - 1),
0683            1 adapter_registers(7))
0684            endif
0685            endif
0686
0687            call linchk (lun,2)
0688
0689            write(lun,8) mf_cs1
0690    8       format(/' ',t8,'MF CS1',t24,z8.8)
0691
0692            if (.not. diagnostic_mode) then
0693
0694            call mba_status_register16_31 (lun,mf_cs1,mf_cs1,0)
0695
0696            call output (lun,mf_cs1,v1mf_cs1,0,0,0,'0')
0697
0698            call linchk (lun,1)
0699
0700            if (dt_function .eq. 20) then
0701
0702            write(lun,10) v1dt_function(dt_function)
0703    10      format(' ',t40,a<compressc (v1dt_function(dt_function))>)
0704
0705            else if (
0706            1 dt_function .ge. 23
0707            1 .and.
0708            1 dt_function .le. 25
0709            1 ) then
0710
0711            write(lun,11) v2dt_function(dt_function)
0712    11      format(' ',t40,a<compressc (v2dt_function(dt_function))>)
0713
0714            else if (
0715            1 (dt_function .eq. 28
0716            1 .or.
0717            1 dt_function .eq. 29)
0718            1 ) then
0719
0720            write(lun,12) v3dt_function(dt_function)
0721    12      format(' ',t40,a<compressc (v3dt_function(dt_function))>)
0722
0723            else if (dt_function .eq. 31) then
0724
0725            write(lun,15) v4dt_function(dt_function)
0726    15      format(' ',t40,a<compressc (v4dt_function(dt_function))>)
0727            endif
0728
```

```
0729            call output (lun,mf_cs1,v2mf_cs1,11,11,11,'0')
0730            endif
0731
0732            call linchk (lun,1)
0733
0734            write(lun,25) mf_is
0735    25      format(' ',t8,'MF IS',t24,z8.8)
0736
0737            if (.not. diagnostic_mode) then
0738
0739            call mba_status_register16_31 (lun,mf_cs1,mf_is,1)
0740
0741            if (dt_cmdaddr .eq. ucb_unit_number
0742            1 .and.
0743            2 dt_function .eq. ucb_function
0744            3 .and.
0745            4 attn_bit_this_tm78 .eq. 0) then
0746
0747            dt_intcode = lib$extzv(0,6,mf_is)
0748
0749            call linchk (lun,1)
0750
0751            if (dt_intcode .ge. 1
0752            1 .and.
0753            2 dt_intcode .le. 4) then
0754
0755            write(lun,30) v1interrupt_code(dt_intcode)
0756    30      format(' ',t40,a<compressc (v1interrupt_code(dt_intcode))>)
0757
0758            else if (dt_intcode .ge. 8
0759            1 .and.
0760            2 dt_intcode .le. 13) then
0761
0762            write(lun,31) v1interrupt_code(dt_intcode)
0763    31      format(' ',t40,a<compressc (v1interrupt_code(dt_intcode))>)
0764
0765            else if (dt_intcode .ge. 16
0766            1 .and.
0767            2 dt_intcode .le. 25) then
0768
0769            write(lun,32) v2interrupt_code(dt_intcode)
0770    32      format(' ',t40,a<compressc (v2interrupt_code(dt_intcode))>)
0771
0772            else if (lib$extzv(14,1,mf_id) .ne. 0) then
0773
0774            write(lun,33) v2interrupt_code(dt_intcode)
0775    33      format(' ',t40,a<compressc (v2interrupt_code(dt_intcode))>)
0776            endif
0777
0778            call output (lun,mf_is,v1mf_is,8,8,8,'0')
0779
0780            dt_fcode = lib$extzv (10,6,mf_is)
0781
0782            call linchk (lun,1)
0783
0784            if ((dt_intcode .eq. 1
0785            1 .or.
```

```
0786            1 dt_intcode .eq. 4
0787            1 .or.
0788            1 dt_intcode .eq. 16
0789            1 .or.
0790            1 dt_intcode .eq. 17)
0791            1 .and.
0792            2 (dt_fcode .ge. 0
0793            3 .and.
0794            4 dt_fcode .le. 1)) then
0795
0796            write(lun,40) fcode_intcode1(dt_fcode)
0797    40      format(' ',t40,a<compressc (fcode_intcode1(dt_fcode))>)
0798
0799            else if (dt_intcode .eq. 2
0800            1 .and.
0801            2 dt_fcode .ge. 1
0802            3 .and.
0803            4 dt_fcode .le. 3) then
0804
0805            write(lun,45) fcode_intcode3(dt_fcode)
0806    45      format(' ',t40,a<compressc (fcode_intcode3(dt_fcode))>)
0807
0808            else if (dt_intcode .eq. 9
0809            1 .and.
0810            2 dt_fcode .ge. 1
0811            3 .and.
0812            4 dt_fcode .le. 3) then
0813
0814            write(lun,50) fcode_intcode9(dt_fcode)
0815    50      format(' ',t40,a<compressc (fcode_intcode9(dt_fcode))>)
0816
0817            else if ((dt_intcode .eq. 18
0818            1 .or.
0819            1 dt_intcode .eq. 19
0820            1 .or.
0821            1 dt_intcode .eq. 20
0822            1 .or.
0823            1 dt_intcode .eq. 21
0824            1 .or.
0825            1 dt_intcode .eq. 22
0826            1 .or.
0827            1 dt_intcode .eq. 23)
0828            1 .and.
0829            2 dt_fcode .ge. 1
0830            3 .and.
0831            4 dt_fcode .le. 11) then
0832
0833            write(lun,60) fcode_intcode18(dt_fcode)
0834    60      format(' ',t40,a<compressc (fcode_intcode18(dt_fcode))>)
0835
0836            else if (dt_intcode .eq. 24
0837            1 .and.
0838            2 (dt_fcode .ge. 1
0839            3 .and.
0840            4 dt_fcode .le. 19)) then
0841
0842            write(lun,65) dt_fcode
```

```
0843    65      format(' ',t40,'FAILURE CODE = ',o2.2,' (OCTAL)')
0844
0845            else if (dt_intcode .eq. 25
0846            1 .and.
0847            2 (dt_fcode .ge. 1
0848            3 .and.
0849            4 dt_fcode .le. 15)) then
0850
0851            write(lun,65) dt_fcode
0852
0853            else if (dt_intcode .eq. 25
0854            1 .and.
0855            2 (dt_fcode .ge. 17
0856            3 .and.
0857            4 dt_fcode .le. 35)) then
0858
0859            write(lun,65) dt_fcode
0860
0861            else if (dt_intcode .eq. 28
0862            1 .and.
0863            2 dt_fcode .ge. 1
0864            3 .and.
0865            4 dt_fcode .le. 2) then
0866
0867            write(lun,75) fcode_intcode28(dt_fcode)
0868    75      format(' ',t40,a<compressc (fcode_intcode28(dt_fcode))>)
0869            endif
0870            endif
0871            endif
0872
0873            call linchk (lun,1)
0874
0875            write(lun,85) mf_tc
0876    85      format(' ',t8,'MF TC',t24,z8.8)
0877
0878            if (.not. diagnostic_mode) then
0879
0880            call mba_status_register16_31 (lun,mf_is,mf_tc,1)
0881
0882            if (dt_cmdaddr .eq. ucb_unit_number
0883            1 .and.
0884            2 dt_function .eq. ucb_function
0885            3 .and.
0886            4 attn_bit_this_tm78 .eq. 0) then
0887
0888            call linchk (lun,1)
0889
0890            write(lun,90) dt_cmdaddr
0891    90      format(' ',t40,'DATA TRANSFER CMD ADDR UNIT = ',
0892            1 i<compress4 (dt_cmdaddr)>,'.')
0893
0894            field = lib$extzv (2,6,mf_tc)
0895
0896            call linchk (lun,1)
0897
0898            write(lun,95) field
0899    95      format(' ',t40,'RECORDS REMAINING = ',i<compress4 (field)>,'.')
```

```
0900
0901            field = lib$extzv (8,4,mf_tc)
0902
0903            call linchk (lun,1)
0904
0905            write(lun,100) field
0906    100     format(' ',t40,'SKIP COUNT = ',i<compress4 (field)>,'.')
0907
0908            field = lib$extzv (12,3,mf_tc)
0909
0910            if (field .ge. 0
0911            1 .and.
0912            2 field .le. 6) then
0913
0914            call linchk (lun,1)
0915
0916            write(lun,105) mf_tc_format(field)
0917    105     format(' ',t40,'FORMAT = ',a<compressc (mf_tc_format(field))>)
0918            endif
0919
0920            call output (lun,mf_tc,v1mf_tc,15,15,15,'0')
0921            endif
0922            endif
0923
0924            call linchk (lun,1)
0925
0926            write(lun,115) mf_mr1
0927    115     format(' ',t8,'MF MR1',t24,z8.8)
0928
0929            if (.not. diagnostic_mode) then
0930
0931            call mba_status_register16_31 (lun,mf_tc,mf_mr1,1)
0932            endif
0933
0934            call linchk (lun,1)
0935
0936            write(lun,120) mf_ab
0937    120     format(' ',t8,'MF AB',t24,z8.8)
0938
0939            if (.not. diagnostic_mode) then
0940
0941            call mba_status_register16_31 (lun,mf_mr1,mf_ab,1)
0942
0943            do 128,i = 0,7
0944
0945            if (lib$extzv(i,1,mf_ab) .eq. 1) then
0946
0947            call linchk (lun,1)
0948
0949            write(lun,125) i
0950    125     format(' ',t40,'ATTENTION MASSBUS UNIT ',i1,'.')
0951            endif
0952
0953    128     continue
0954            endif
0955
0956            call linchk (lun,1)
```

```
0957
0958            write(lun,130) mf_bc
0959    130     format(' ',t8,'MF BC',t24,z8.8)
0960
0961            if (.not. diagnostic_mode) then
0962
0963            call mba_status_register16_31 (lun,mf_ab,mf_bc,1)
0964
0965            if (dt_cmdaddr .eq. ucb_unit_number
0966            1 .and.
0967            2 dt_function .eq. ucb_function
0968            3 .and.
0969            4 attn_bit_this_tm78 .eq. 0) then
0970
0971            field = lib$extzv (0,16,mf_bc)
0972
0973            call linchk (lun,1)
0974
0975            write(lun,135) field
0976    135     format(' ',t40,'BYTE COUNT = ',i<compress4 (field)>,'.')
0977            endif
0978            endif
0979
0980            call linchk (lun,1)
0981
0982            write(lun,140) mf_dt
0983    140     format(' ',t8,'MF DT',t24,z8.8)
0984
0985            if (.not. diagnostic_mode) then
0986
0987            call mba_status_register16_31 (lun,mf_bc,mf_dt,1)
0988
0989            field = lib$extzv (0,8,mf_dt)
0990
0991            if (field .eq. 65) then
0992
0993            call linchk (lun,1)
0994
0995            write(lun,145)
0996    145     format(' ',t40,'DRIVE TYPE TU78')
0997            endif
0998
0999            call output (lun,mf_dt,v1mf_dt,10,10,11,'0')
1000
1001            call output (lun,mf_dt,v2mf_dt,14,14,15,'0')
1002            endif
1003
1004            call linchk (lun,1)
1005
1006            write(lun,155) mf_ds
1007    155     format(' ',t8,'MF DS',t24,z8.8)
1008
1009            if (.not. diagnostic_mode) then
1010
1011            call mba_status_register16_31 (lun,mf_dt,mf_ds,1)
1012
1013            call output (lun,mf_ds,v1mf_ds,4,4,4,'0')
```

```
1014
1015            call output (lun,mf_ds,v2mf_ds,6,6,15,'0')
1016            else
1017
1018            call linchk (lun,1)
1019
1020            write(lun,157) 'DAIGNOSTIC MODE'
1021     157    format(' ',t40,a)
1022            endif
1023
1024            call linchk (lun,1)
1025
1026            write(lun,160) mf_sn
1027     160    format(' ',t8,'MF SN',t24,z8.8)
1028
1029            if (.not. diagnostic_mode) then
1030
1031            call mba_status_register16_31 (lun,mf_ds,mf_sn,1)
1032            endif
1033
1034            call linchk (lun,1)
1035
1036            write(lun,165) mf_mr2
1037     165    format(' ',t8,'MF MR2',t24,z8.8)
1038
1039            if (.not. diagnostic_mode) then
1040
1041            call mba_status_register16_31 (lun,mf_sn,mf_mr2,1)
1042            endif
1043
1044            call linchk (lun,1)
1045
1046            write(lun,170) mf_mr3
1047     170    format(' ',t8,'MF MR3',t24,z8.8)
1048
1049            if (.not. diagnostic_mode) then
1050
1051            call mba_status_register16_31 (lun,mf_mr2,mf_mr3,1)
1052            endif
1053
1054            ndt_cmdaddr = lib$extzv (8,2,mf_ndta)
1055
1056            call linchk (lun,1)
1057
1058            write(lun,175) mf_ndta
1059     175    format(' ',t8,'MF NDTA',t24,z8.8)
1060
1061            if (.not. diagnostic_mode) then
1062
1063            call mba_status_register16_31 (lun,mf_mr3,mf_ndta,1)
1064
1065            ndt_function = lib$extzv (1,5,mf_ndt(ndt_cmdaddr))
1066
1067            if (
1068            1 attn_bit_this_tm78 .eq. 1
1069            1 .or.
1070            1 (ndt_cmdaddr .eq. ucb_unit_numbe
```

```
1071            1 .and.
1072            1 ndt_function .eq. ucb_function)
1073            1 .or.
1074            1 ucb$l_mf_cmd .lt. 0
1075            1 ) then
1076
1077            dt_function = 0
1078
1079            ndt_intcode = lib$extzv (0,6,mf_ndta)
1080
1081            call linchk (lun,1)
1082
1083            if (ndt_intcode .ge. 1
1084            1 .and.
1085            2 ndt_intcode .le. 13) then
1086
1087            write(lun,180) v1interrupt_code(ndt_intcode)
1088     180    format(' ',t40,a<compressc (v1interrupt_code(ndt_intcode))>)
1089
1090            else if (ndt_intcode .ge. 23
1091            1 .and.
1092            2 ndt_intcode .le. 25) then
1093
1094            write(lun,181) v2interrupt_code(ndt_intcode)
1095     181    format(' ',t40,a<compressc (v2interrupt_code(ndt_intcode))>)
1096            endif
1097
1098            ndt_fcode = lib$extzv (8,2,mf_ndta)
1099
1100            call linchk (lun,1)
1101
1102            write(lun,190) ndt_fcode
1103     190    format(' ',t40,'ATTENTION CMD ADDR UNIT = ',i1,'.')
1104
1105            ndt_fcode = lib$extzv (10,6,mf_ndta)
1106
1107            call linchk (lun,1)
1108
1109            if ((ndt_intcode .eq. 1
1110            1 .or.
1111            1 ndt_intcode .eq. 4
1112            1 .or.
1113            1 ndt_intcode .eq. 16
1114            1 .or.
1115            1 ndt_intcode .eq. 17)
1116            1 .and.
1117            2 (ndt_fcode .ge. 0
1118            3 .and.
1119            4 ndt_fcode .le. 1)) then
1120
1121            write(lun,195) fcode_intcode1(ndt_fcode)
1122     195    format(' ',t40,a<compressc (fcode_intcode1(ndt_fcode))>)
1123
1124            else if (ndt_intcode .eq. 3
1125            1 .and.
1126            2 ndt_fcode .ge. 1
1127            3 .and.
```

```
1128            4 ndt_fcode .le. 3) then
1129
1130            write(lun,200) fcode_intcode3(ndt_fcode)
1131      200   format(' ',t40,a<compressc (fcode_intcode3(ndt_fcode))>)
1132
1133            else if (ndt_intcode .eq. 9
1134            1 .and.
1135            2 ndt_fcode .ge. 1
1136            3 .and.
1137            4 ndt_fcode .le. 3) then
1138
1139            write(lun,205) fcode_intcode9(ndt_fcode)
1140      205   format(' ',t40,a<compressc (fcode_intcode9(ndt_fcode))>)
1141
1142            else if (ndt_intcode .eq. 13
1143            1 .and.
1144            2 ndt_fcode .ge. 1
1145            3 .and.
1146            4 ndt_fcode .le. 4) then
1147
1148            write(lun,210) fcode_intcode13(ndt_fcode)
1149      210   format(' ',t40,a<compressc (fcode_intcode13(ndt_fcode))>)
1150
1151            else if ((ndt_intcode .eq. 19
1152            1 .or.
1153            1 ndt_intcode .eq. 20
1154            1 .or.
1155            1 ndt_intcode .eq. 21
1156            1 .or.
1157            1 ndt_intcode .eq. 22
1158            1 .or.
1159            1 ndt_intcode .eq. 23)
1160            1 .and.
1161            2 ndt_fcode .ge. 1
1162            3 .and.
1163            4 ndt_fcode .le. 11) then
1164
1165            write(lun,215) fcode_intcode18(ndt_fcode)
1166      215   format(' ',t40,a<compressc (fcode_intcode18(ndt_fcode))>)
1167
1168            else if (ndt_intcode .eq. 24
1169            1 .and.
1170            2 (ndt_fcode .ge. 1
1171            3 .and.
1172            4 ndt_fcode .le. 19)) then
1173
1174            write(lun,65) ndt_fcode
1175
1176            else if (ndt_intcode .eq. 25
1177            1 .and.
1178            2 (ndt_fcode .ge. 1
1179            3 .and.
1180            4 ndt_fcode .ge. 15)) then
1181
1182            write(lun,65) ndt_fcode
1183
1184            else if (ndt_intcode .eq. 25
```

```
1185            1 .and.
1186            2 (ndt_fcode .ge. 17
1187            3 .and.
1188            4 ndt_fcode .le. 35)) then
1189
1190            write(lun,65) ndt_fcode
1191
1192            else if (ndt_intcode .eq. 28
1193            1 .and.
1194            2 ndt_fcode .ge. 1
1195            3 .and.
1196            4 ndt_fcode .le. 2) then
1197
1198            write(lun,230) fcode_intcode28(ndt_fcode)
1199    230     format(' ',t40,a<compressc (fcode_intcode28(ndt_fcode))>)
1200            else
1201
1202            write(lun,231) ndt_intcode
1203    231     format(' ',t40,''NDT'' INTERRUPT CODE ',o2.2,' (OCTAL)')
1204
1205            call linchk (lun,1)
1206
1207            write(lun,232) ndt_fcode
1208    232     format(' ',t40,''NDT'' FAILURE CODE ',o2.2,' (OCTAL)')
1209            endif
1210            endif
1211            endif
1212
1213            do 250,i = 0,3
1214
1215            call linchk (lun,1)
1216
1217            write(lun,235) i,mf_ndt(i)
1218    235     format(' ',t8,'MF NDT',i1,t24,z8.8)
1219
1220            if (.not. diagnostic_mode) then
1221
1222            call mba_status_register16_31 (lun,mf_ndt(max(0,i-1)),mf_ndt(i),1)
1223
1224            if (i .eq. ucb_unit_number
1225            1 .and.
1226            4 ndt_function .eq. ucb_function) then
1227
1228            call linchk (lun,1)
1229
1230            if (ndt_function .ge. 1
1231            1 .and.
1232            2 ndt_function .le. 19) then
1233
1234            write(lun,240) vlndt_function(ndt_function)
1235    240     format(' ',t40,a<compressc (vlndt_function(ndt_function))>)
1236            endif
1237
1238            field = lib$extzv (8,8,mf_ndt(i))
1239
1240            call linchk (lun,1)
1241
```

```
1242            write(lun,245) field
1243    245     format(' ',t40,'COMMAND COUNT = ',i<compress4 (field)>,'.')
1244            endif
1245            endif
1246
1247    250     continue
1248
1249            call linchk (lun,1)
1250
1251            write(lun,260) mf_id
1252    260     format(' ',t8,'MF ID',t24,z8.8)
1253
1254            if (.not. diagnostic_mode) then
1255
1256            call mba_status_register16_31 (lun,mf_ndt(3),mf_id,1)
1257
1258            call output (lun,mf_id,v1mf_id,8,8,15,'0')
1259
1260            if (emb$w_hd_entry .ne. 96
1261            1 .and.
1262            1 ((dt_cmdaddr .eq. ucb_unit_number
1263            1 .and.
1264            2 dt_function .eq. ucb_function
1265            3 .and.
1266            3 dt_intcode .ne. 17
1267            3 .and.
1268            4 dt_fcode .ne. 0)
1269            5 .or.
1270            6 (ndt_cmdaddr .eq. ucb_unit_number
1271            7 .and.
1272            8 ndt_function .eq. ucb_function
1273            9 .and.
1274            1 ndt_intcode .ne. 5
1275            1 .and.
1276            1 ndt_intcode .ne. 6
1277            1 .and.
1278            1 ndt_intcode .ne. 7
1279            1 .and.
1280            1 ndt_fcode .ne. 0))
1281            1 .or.
1282            1 ucb$l_mf_cmd .lt. 0) then
1283
1284            call linchk (lun,3)
1285
1286            write(lun,265)
1287    265     format(/' ','EXTENDED SENSE INFORMATION',/)
1288
1289            call linchk (lun,8)
1290
1291            write(lun,270) (mf_exsns(i),i = 1,5)
1292    270     format(' ',t8,'BYTE 1',t30,z2.2,/,t40,'COMMAND CODE',/,
1293            1 t8,'BYTE 2',t30,z2.2,/,t40,'INTERRUPT CODE',/,
1294            2 t8,'BYTE 3',t30,z2.2,/,t40,'FAILURE CODE',/,
1295            3 t8,'RPFAIL',t30,z2.2,/,
1296            4 t8,'RPATH',t30,z2.2)
1297
1298            call output (lun,mf_exsns(5),v1mf_exsns5,0,0,7,'0')
```

```
1299
1300            call linchk (lun,2)
1301
1302            write(lun,295) (mf_exsns(6),i = 1,2)
1303    295     format(' ',t8,'RSTAT',t30,z2.2,/,t40,
1304            1 'RMC STATUS = ',o3.3,' (OCTAL)')
1305
1306            call linchk (lun,7)
1307
1308            write(lun,300) (mf_exsns(i),i = 7,13)
1309    300     format(' ',t8,'RCMCP',t30,z2.2,/,
1310            1 t8,'RAMT',t30,z2.2,/,
1311            2 t8,'RDON',t30,z2.2,/,
1312            3 t8,'RILL',t30,z2.2,/,
1313            4 t8,'RMK2',t30,z2.2,/,
1314            5 t8,'EMK',t30,z2.2,/,
1315            6 t8,'RPSTA',t30,z2.2)
1316
1317            call output (lun,mf_exsns(13),v1mf_exsns13,0,0,7,'0')
1318
1319            call linchk (lun,5)
1320
1321            write(lun,335) (mf_exsns(i),i = 14,18)
1322    335     format(' ',t8,'RPOSTN',t30,z2.2,/,
1323            1 t8,'RDATA',t30,z2.2,/,
1324            2 t8,'CRCWRD',t30,z2.2,/,
1325            3 t8,'ECCOR',t30,z2.2,/,
1326            4 t8,'ECCSTA',t30,z2.2)
1327
1328            call output (lun,mf_exsns(18),v1mf_exsns18,0,0,7,'0')
1329
1330            do 365,i = 0,7
1331
1332            call linchk (lun,1)
1333
1334            write(lun,360) i,mf_exsns(19 + i)
1335    360     format(' ',t8,'CH',T1,'TIE',t30,z2.2)
1336
1337    365     continue
1338
1339            call linchk (lun,4)
1340
1341            write(lun,370) (mf_exsns(i),i = 27,30)
1342    370     format(' ',t8,'CHPTIE',t30,z2.2,/,
1343            1 t8,'RTIER',t30,z2.2,/,
1344            2 t8,'TAMT',t30,z2.2,/,
1345            3 t8,'PSTAT',t30,z2.2)
1346
1347            call output (lun,mf_exsns(30),v1mf_exsns30,0,0,7,'0')
1348
1349            call linchk (lun,2)
1350
1351            write(lun,390) (mf_exsns(i),i = 31,32)
1352    390     format(' ',t8,'PRDD',t30,z2.2,/,
1353            1 t8,'CASSTA',t30,z2.2)
1354
1355            field = lib$extzv(0,3,mf_exsns(32))
```

```
1356
1357            call linchk (lun,1)
1358
1359            write(lun,400) field
1360    400     format(' ',t40,'DRIVE SELECT = ',i<compress4 (field)>,'.')
1361
1362            call output (lun,mf_exsns(32),v1mf_exsns32,3,3,7,'0')
1363
1364            call linchk (lun,1)
1365
1366            write(lun,410) mf_exsns(33)
1367    410     format(' ',t8,'CBUSSTA',t30,z2.2)
1368
1369            call output (lun,mf_exsns(33),v1mf_exsns33,0,0,7,'0')
1370
1371            call linchk (lun,1)
1372
1373            write(lun,415) mf_exsns(34)
1374    415     format(' ',t8,'DBUSSTA',t30,z2.2)
1375
1376            call output (lun,mf_exsns(34),v1mf_exsns34,0,0,7,'0')
1377
1378            call linchk (lun,1)
1379
1380            write(lun,420) mf_exsns(35)
1381    420     format(' ',t8,'WMCSTA',t30,z2.2)
1382
1383            field = lib$extzv (7,1,mf_exsns(36))
1384
1385            do 430,i = 0,field
1386
1387            call linchk (lun,2)
1388
1389            write(lun,422) i,mf_exsns(36 + i)
1390    422     format(' ',t8,'TUSEL ',i1,t30,z2.2)
1391
1392            field = lib$extzv (0,2,mf_exsns(36 + i))
1393
1394            write(lun,424) field
1395    424     format(' ',t40,'TAPE UNIT SELECT = ',i<compress4 (field)>,'.')
1396
1397            do 430,k = 2,5
1398
1399            field = lib$extzv (k,1,mf_exsns(36 + i))
1400
1401            if (field .ne. 0) then
1402
1403            call linchk (lun,1)
1404
1405            write(lun,426) v1tu_selx(k,i)
1406    426     format(' ',t40,'TU PORT ',a<compressc (v1tu_selx(k,i))>,
1407            1 ' PATH ENABLED')
1408            endif
1409
1410    430     continue
1411
1412            call linchk (lun,1)
```

```
1413
1414            write(lun,435) mf_exsns(38)
1415     435    format(' ',t8,'WRTDAT',t30,z2.2)
1416
1417            call linchk (lun,3)
1418
1419            write(lun,450) ((mf_exsns(39 + j + j),i = 1,0,-1),j = 0,4,2)
1420     450    format(' ',t8,'BYTCNT',t28,2z2.2,/,
1421            1 t8,'PADCNT',t28,2z2.2,/,
1422            2 t8,'ERRCNT',t28,2z2.2)
1423
1424            call linchk (lun,3)
1425
1426            write(lun,455) (mf_exsns(45 + i),i = 0,2)
1427     455    format(' ',t8,'DDR A',t30,z2.2,/,t8,'DDR B',t30,z2.2,/,
1428            1 t8,'DDR C',t30,z2.2)
1429
1430            call output (lun,mf_exsns(47),v1mf_exsns47,3,3,7,'0')
1431
1432            call linchk (lun,1)
1433
1434            write(lun,465) mf_exsns(48)
1435     465    format(' ',t8,'INSTA',t30,z2.2)
1436
1437            call output (lun,mf_exsns(48),v1mf_exsns48,0,0,0,'0')
1438
1439            if (dt_function .eq. 24) then
1440
1441            call output (lun,mf_exsns(48),v1mf_exsns48,0,1,1,'0')
1442            endif
1443
1444            call output (lun,mf_exsns(48),v1mf_exsns48,0,2,7,'0')
1445
1446            call linchk (lun,1)
1447
1448            write(lun,470) mf_exsns(49)
1449     470    format(' ',t8,'MTA 0',t30,z2.2)
1450
1451            call output (lun,mf_exsns(49),v1mf_exsns49,0,0,7,'0')
1452
1453            call linchk (lun,1)
1454
1455            write(lun,475) mf_exsns(50)
1456     475    format(' ',t8,'MTA 1',t30,z2.2)
1457
1458            call output (lun,mf_exsns(50),v1mf_exsns50,0,0,7,'0')
1459
1460            call linchk (lun,1)
1461
1462            write(lun,480) mf_exsns(51)
1463     480    format(' ',t8,'MTA 2',t30,z2.2)
1464
1465            field = lib$extzv (0,3,mf_exsns(51))
1466
1467            call linchk (lun,1)
1468
1469            if (field .eq. 6) then
```

```
1470
1471            write(lun,485) 'A'
1472     485    format(' ',t40,'PORT SELECT = MASSBUS ',a)
1473
1474            else if (field .eq. 5) then
1475
1476            write(lun,485) 'B'
1477
1478            else if (field .eq. 3) then
1479
1480            write(lun,485) 'A/B'
1481            else
1482
1483            write(lun,490)
1484     490    format(' ',t40,'NEITHER MASSBUS SELECTED')
1485            endif
1486
1487            call linchk (lun,1)
1488
1489            if (lib$extzv(3,2,mf_exsns(51)) .eq. 2) then
1490
1491            write(lun,495)
1492     495    format(' ',t40,'125 IPS TRANSPORT')
1493            else
1494
1495            write(lun,500)
1496     500    format(' ',t40,'NOT TU78 SPEED')
1497            endif
1498
1499            call output (lun,mf_exsns(51),v1mf_exsns51,5,5,7,'0')
1500
1501            do 515,i = 3,4
1502
1503            call linchk (lun,1)
1504
1505            write(lun,510) i,mf_exsns(49 + i)
1506     510    format(' ',t8,'MTA ',i1,t30,z2.2)
1507
1508     515    continue
1509
1510            call linchk (lun,1)
1511
1512            write(lun,520) (mf_exsns(i),i = 52,53)
1513     520    format(' ',t40,'SERIAL NUMBER = ',z2.2,z2.2)
1514
1515            call linchk (lun,1)
1516
1517            write(lun,525) mf_exsns(54)
1518     525    format(' ',t8,'MTA 5',t30,z2.2)
1519
1520            call linchk (lun,1)
1521
1522            write(lun,530) lib$extzv(0,2,mf_exsns(54))
1523     530    format(' ',t40,'THRESHOLD = ',i1,'.')
1524
1525            call output (lun,mf_exsns(54),v1mf_exsns54,2,2,7,'0')
1526
```

```
1527            field = lib$extzv (0,8,mf_exsns(55))
1528
1529            call linchk (lun,2)
1530
1531            write(lun,535) mf_exsns(55),field
1532     535    format(' ',t8,'RETCNT',t30,z2.2,/,t40,
1533            1 'RETRY COUNT = ',i<compress4 (field)>,'.')
1534
1535            call linchk (lun,1)
1536
1537            write(lun,540) mf_exsns(56)
1538     540    format(' ',t8,'RETCNT+1',t30,z2.2)
1539
1540            if (field .ne. 0) then
1541
1542            call output (lun,mf_exsns(56),v1mf_exsns56,5,5,7,'0')
1543            endif
1544
1545            call linchk (lun,1)
1546
1547            write(lun,545) mf_exsns(57)
1548     545    format(' ',t8,'TUX',t30,z2.2)
1549
1550            call output (lun,mf_exsns(57),v1mf_exsns57,0,0,7,'0')
1551
1552            call linchk (lun,1)
1553
1554            write(lun,550) mf_exsns(58)
1555     550    format(' ',t8,'XFRCTL',t30,z2.2)
1556
1557            call linchk (lun,1)
1558
1559            write(lun,570) mf_exsns(59)
1560     570    format(' ',t8,'XRETRY',t30,z2.2)
1561
1562            call linchk (lun,1)
1563
1564            write(lun,575) mf_exsns(60)
1565     575    format(' ',t8,'ENAON',t30,z2.2)
1566
1567            if (mf_exsns(60) .ne. 0) then
1568
1569            call linchk (lun,1)
1570
1571            write(lun,580)
1572     580    format(' ',t40,'KEYPAD ENABLED')
1573            endif
1574            endif
1575            endif
1576
1577            if (ucb$l_mf_cmd .ge. 0) then
1578
1579            if (emb$w_hd_entry .ne. 98) then
1580
1581            call linchk (lun,1)
1582
1583            write(lun,585)
```

```
ML1
033
033
033
033
033
033
033
033
033
033
034
034
034
034
034
034
034
034
034
034
035
035
035
035
035
035
035
035
035
035
036
036
036
036
036
036
036
036
036
036
037
037
037
037
037
037
037
037
037
037
038
038
038
038
038
038
038
038
```

```
1584     585     format(' ',:)
1585
1586             if (emb$w_hd_entry .ne. 98) then
1587
1588             call ucb$b_ertcnt (lun,emb$b_dv_ertcnt)
1589
1590             call ucb$b_ertmax (lun,emb$b_dv_ertmax)
1591             endif
1592
1593             call orb$l_owner (lun,emb$l_dv_ownuic)
1594
1595             call ucb$l_char (lun,emb$l_dv_char)
1596
1597             call ucb$w_sts (lun,emb$w_dv_sts)
1598
1599             call ucb$l_opcnt (lun,emb$l_dv_opcnt)
1600
1601             call ucb$w_errcnt (lun,emb$w_dv_errcnt)
1602
1603             if (emb$w_hd_entry .ne. 98) then
1604
1605             call linchk (lun,1)
1606
1607             write(lun,585)
1608
1609             call mftape_qio (lun,emb$w_dv_func)
1610
1611             call irp$w_bcnt (lun,emb$w_dv_bcnt)
1612
1613             call irp$w_boff (lun,emb$w_dv_boff)
1614
1615             call irp$l_pid (lun,emb$l_dv_rqpid)
1616
1617             call irp$q_iosb (lun,emb$l_dv_iosb1)
1618             endif
1619             endif
1620             endif
1621
1622             return
1623             end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	8218	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	2308	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	8192	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	19230	

ENTRY POINTS

Address	Type	Name
0-00000000		MFTAPE

VARIABLES

Address	Type	Name
2-00001494	I*4	ATTN_BIT_THIS_TM78
2-0000147C	I*4	DT_CMDADDR
2-0000148C	I*4	DT_FUNCTION
3-0000001C	L*1	EMB$B_DV_CLASS
3-00000011	L*1	EMB$B_DV_ERTMAX
3-0000003A	L*1	EMB$B_DV_SLAVE
3-00000036	I*4	EMB$L_DV_CHAR
3-00000016	I*4	EMB$L_DV_IOSB2
3-0000004E	I*4	EMB$L_DV_NUMREG
3-00000032	I*4	EMB$L_DV_OWNUIC
3-00000000	I*4	EMB$L_HD_SID
3-00000024	I*2	EMB$W_DV_BCNT
3-0000002C	I*2	EMB$W_DV_ERRCNT
3-0000001A	I*2	EMB$W_DV_STS
3-00000004	I*2	EMB$W_HD_ENTRY
2-00001470	I*4	FIELD
2-000014B0	I*4	J
2-00001498	I*4	LIB$EXTV
3-0000007E	I*4	MF_AB
3-0000006E	I*4	MF_CS1
3-00000086	I*4	MF_DT
3-00000072	I*4	MF_IS
3-00000092	I*4	MF_MR2
3-0000009A	I*4	MF_NDTA
3-00000076	I*4	MF_TC
2-00001478	I*4	NDT_FCODE
2-00001488	I*4	NDT_INTCODE
3-000000B2	I*4	UCB$L_MF_CMD
2-000014A4	I*4	UCB_UNIT_NUMBER
2-0000146F	L*1	DIAGNOSTIC_MODE
2-00001474	I*4	DT_FCODE
2-00001484	I*4	DT_INTCODE
3-00000010	L*1	EMB$B_DV_ERTCNT
3-0000003E	L*1	EMB$B_DV_NAMLNG
3-0000001D	L*1	EMB$B_DV_TYPE
3-00000012	I*4	EMB$L_DV_IOSB1
3-00000026	I*4	EMB$L_DV_MEDIA
3-0000002E	I*4	EMB$L_DV_OPCNT
3-0000001E	I*4	EMB$L_DV_RQPID
3-0000003F	CHAR	EMB$T_DV_NAME
3-00000022	I*2	EMB$W_DV_BOFF
3-0000003C	I*2	EMB$W_DV_FUNC
3-0000002A	I*2	EMB$W_DV_UNIT
3-0000000E	I*2	EMB$W_HD_ERRSEQ
2-000014A8	I*4	I
2-000014AC	I*4	K
AP-00000004@	L*1	LUN
3-00000082	I*4	MF_BC
3-0000008A	I*4	MF_DS
3-000000AE	I*4	MF_ID
3-0000007A	I*4	MF_MR1
3-00000096	I*4	MF_MR3
3-0000008E	I*4	MF_SN
2-00001480	I*4	NDT_CMDADDR
2-00001490	I*4	NDT_FUNCTION
2-000014A0	I*4	SELECTED_MAP_REGISTER
2-0000149C	I*4	UCB_FUNCTION

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000052	I*4	ADAPTER_REGISTERS	28	(7)
3-00000000	L*1	EMB	512	(0:511)
3-00000052	I*4	EMB$L_DV_REGSAV	420	(0:104)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)
2-00000311	CHAR	FCODE_INTCODE1	36	(0:1)
2-000003F2	CHAR	FCODE_INTCODE13	96	(4)
2-00000452	CHAR	FCODE_INTCODE18	308	(11)
2-00000586	CHAR	FCODE_INTCODE28	58	(2)
2-00000335	CHAR	FCODE_INTCODE3	93	(3)
2-00000392	CHAR	FCODE_INTCODE9	96	(3)
3-000000BA	L*1	MF_EXSNS	60	(60)
3-0000009E	I*4	MF_NDT	16	(0:3)
2-00000998	CHAR	MF_TC_FORMAT	189	(0:6)
2-0000080D	CHAR	V1DT_FUNCTION	20	(20:20)
2-00000000	CHAR	V1INTERRUPT_CODE	351	(13)
2-00000B68	CHAR	V1MF_CS1	7	(0:0)
2-000008E2	CHAR	V1MF_DS	32	(4:4)
2-00000888	CHAR	V1MF_DT	46	(10:11)
2-00000C2F	CHAR	V1MF_EXSNS13	136	(0:7)
2-00000CB7	CHAR	V1MF_EXSNS18	240	(0:7)
2-00000DA7	CHAR	V1MF_EXSNS30	168	(0:7)
2-00000E4F	CHAR	V1MF_EXSNS32	90	(3:7)
2-00000EA9	CHAR	V1MF_EXSNS33	120	(0:7)
2-00000F21	CHAR	V1MF_EXSNS34	168	(0:7)
2-00000FC9	CHAR	V1MF_EXSNS47	145	(3:7)
2-0000105A	CHAR	V1MF_EXSNS48	264	(0:7)
2-00001162	CHAR	V1MF_EXSNS49	104	(0:7)
2-00000B7F	CHAR	V1MF_EXSNS5	176	(0:7)
2-000011CA	CHAR	V1MF_EXSNS50	112	(0:7)
2-0000123A	CHAR	V1MF_EXSNS51	63	(5:7)
2-00001279	CHAR	V1MF_EXSNS54	120	(2:7)
2-000012F1	CHAR	V1MF_EXSNS56	78	(5:7)
2-0000133F	CHAR	V1MF_EXSNS57	240	(0:7)
2-00000A55	CHAR	V1MF_ID	232	(8:15)
2-00000B3D	CHAR	V1MF_IS	14	(8:8)
2-00000B4B	CHAR	V1MF_TC	29	(15:15)
2-000005C0	CHAR	V1NDT_FUNCTION	589	(19)
2-0000142F	CHAR	V1TU_SELX	64	(2:5, 0:1)
2-00000821	CHAR	V2DT_FUNCTION	60	(23:25)
2-0000015F	CHAR	V2INTERRUPT_CODE	434	(15:28)
2-00000B6F	CHAR	V2MF_CS1	16	(11:11)
2-00000902	CHAR	V2MF_DS	150	(6:15)
2-000008B6	CHAR	V2MF_DT	44	(14:15)
2-0000085D	CHAR	V3DT_FUNCTION	30	(28:29)
2-0000087B	CHAR	V4DT_FUNCTION	13	(31:31)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-0000005A	8'	1-0000006E	10'	1-0000007A	11'	1-00000086	12'	1-00000092	15'	1-0000009E	25'
1-000000B0	30'	1-000000BC	31'	1-000000C8	32'	1-000000D4	33'	1-000000E0	40'	1-000000EC	45'
1-000000F8	50'	1-00000104	60'	1-00000110	65'	1-00000134	75'	1-00000140	85'	1-00000152	90'
1-00000181	95'	1-000001A6	100'	1-000001C4	105'	1-000001DB	115'	1-000001EE	120'	1-00000200	125'
**	128	1-00000224	130'	1-00000236	135'	1-00000254	140'	1-00000266	145'	1-0000027D	155'
1-0000028F	157'	1-00000296	160'	1-000002A8	165'	1-000002BB	170'	1-000002CE	175'	1-000002E2	180'
1-000002EE	181'	1-000002FA	190'	1-00000321	195'	1-0000032D	200'	1-00000339	205'	1-00000345	210'
1-00000351	215'	1-0000035D	230'	1-00000369	231'	1-00000393	232'	1-000003BB	235'	1-000003D0	240'
1-000003DC	245'	**	250	1-000003FD	260'	1-0000040F	265'	1-00000431	270'	1-000004B8	295'
1-000004E9	300'	1-0000054F	335'	1-000005A0	360'	**	365	1-000005B6	370'	1-000005F5	390'
1-00000616	400'	1-00000636	410'	1-0000064A	415'	1-0000065E	420'	1-00000671	422'	1-00000686	424'
1-000006AA	426'	**	430	1-000006CF	435'	1-000006E2	450'	1-0000071B	455'	1-0000074B	465'
1-0000075D	470'	1-0000076F	475'	1-00000781	480'	1-00000793	485'	1-000007B2	490'	1-000007D2	495'
1-000007EB	500'	1-00000801	510'	**	515	1-00000814	520'	1-00000832	525'	1-00000844	530'
1-0000085D	535'	1-0000088C	540'	1-000008A1	545'	1-000008B1	550'	1-000008C4	570'	1-000008D7	575'
1-000008E9	580'	1-000008FF	585'								

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
I*4	COMPRESS4
	FRCTOF
	IRP$W_BCNT
	LINCHR
	MBA_STATUS_REGISTER16_31
	OUTPUT
	UCB$L_CHAR
	UCB$W_STS
I*4	COMPRESSC
	IRP$L_PID
	IRP$W_BOFF
	MBA_CONTROL_REGISTERS
	MFTAPE_QIO
	UCB$B_ERTCNT
	UCB$L_OPCNT
	DHEAD1
	IRP$Q_IOSB
I*4	LIB$EXTZV
	MBA_MAPPING_REGISTER
	ORB$L_OWNER
	UCB$B_ERTMAX
	UCB$W_ERRCNT

```
0001
0002
0003
0004          Subroutine MFTAPE_QIO (lun,emb$w_dv_func)
0005          include 'src$:qiocommon.for /nolist'
0270
0271
0272          byte            lun
0273
0274          integer*2       emb$w_dv_func
0275
0276          integer*4       qiocode(0:1,0:63)
0277
0278
0279          if (qiocode(0,0) .eq. 0) then
0280
0281          qiocode(1,00) = %loc(io$_nop)
0282
0283          qiocode(1,01) = %loc(io$_unload)
0284
0285          qiocode(1,02) = %loc(io$_spacefile)
0286
0287          qiocode(1,03) = %loc(io$_recal)
0288
0289          qiocode(1,04) = %loc(io$_drvclr)
0290
0291          qiocode(1,06) = %loc(io$_erasetape)
0292
0293          qiocode(1,08) = %loc(io$_packack)
0294
0295          qiocode(1,09) = %loc(io$_spacerecord)
0296
0297          qiocode(1,10) = %loc(io$_writecheck)
0298
0299          qiocode(1,11) = %loc(io$_writepblk)
0300
0301          qiocode(1,12) = %loc(io$_readpblk)
0302
0303          qiocode(1,25) = %loc(io$_readpreset)
0304
0305          qiocode(1,26) = %loc(io$_setchar)
0306
0307          qiocode(1,27) = %loc(io$_sensechar)
0308
0309          qiocode(1,28) = %loc(io$_writemark)
0310
0311          qiocode(1,30) = %loc(io$_clean)
0312
0313          qiocode(1,32) = %loc(io$_writelblk)
0314
0315          qiocode(1,33) = %loc(io$_readlblk)
0316
0317          qiocode(1,34) = %loc(io$_rewindoff)
0318
0319          qiocode(1,35) = %loc(io$_setmode)
0320
```

```
0321            qiocode(1,36) = %loc(io$_rewind)
0322
0323            qiocode(1,37) = %loc(io$_skipfile)
0324
0325            qiocode(1,38) = %loc(io$_skiprecord)
0326
0327            qiocode(1,39) = %loc(io$_sensemode)
0328
0329            qiocode(1,40) = %loc(io$_writeof)
0330
0331            qiocode(1,48) = %loc(io$_writevblk)
0332
0333            qiocode(1,49) = %loc(io$_readvblk)
0334
0335            qiocode(1,50) = %loc(io$_access)
0336
0337            qiocode(1,51) = %loc(io$_create)
0338
0339            qiocode(1,52) = %loc(io$_deaccess)
0340
0341            qiocode(1,53) = %loc(io$_delete)
0342
0343            qiocode(1,54) = %loc(io$_modify)
0344
0345            qiocode(1,56) = %loc(io$_acpcontrol)
0346
0347            qiocode(1,57) = %loc(io$_mount)
0348
0349            do 10,i = 0,63
0350
0351            qiocode(0,i) = 33
0352
0353            if (qiocode(1,i) .eq. 0) then
0354
0355            qiocode(1,i) = %loc(qio_string)
0356            endif
0357
0358      10    continue
0359            endif
0360
0361            call irp$w_func (lun,emb$w_dv_func,
0362            1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0363
0364            return
0365
0366            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	324	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	548	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	QIOCOMMON	1247	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	2127	

ENTRY POINTS

Address	Type	Name
0-00000000		MFTAPE_QIO

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008a	I*2	EMBSW_DV_FUNC	2-00000200	I*4	I
3-00000442	CHAR	IO$_ABORT	3-0000034D	CHAR	IO$_ACCESS
3-000003C2	CHAR	IO$_ACPCONTROL	3-000004B3	CHAR	IO$_AVAILABLE
3-00000297	CHAR	IO$_CLEAN	3-00000369	CHAR	IO$_CREATE
3-00000385	CHAR	IO$_DEACCESS	3-00000393	CHAR	IO$_DELETE
3-0000026D	CHAR	IO$_DIAGNOSE	3-00000065	CHAR	IO$_DRVCLR
3-000004CB	CHAR	IO$_DSE	3-000000A9	CHAR	IO$_ERASETAPE
3-00000276	CHAR	IO$_FORMAT	3-00000071	CHAR	IO$_INITIALIZE
3-00000014	CHAR	IO$_LOADMCODE	3-000003A1	CHAR	IO$_MODIFY
3-000003E2	CHAR	IO$_MOUNT	3-00000000	CHAR	IO$_NOP
3-0000009D	CHAR	IO$_OFFSET	3-000000EB	CHAR	IO$_PACKACK
3-000000E0	CHAR	IO$_QSTOP	3-000003EF	CHAR	IO$_RDSTATS
3-00000421	CHAR	IO$_READCSR	3-00000169	CHAR	IO$_READHEAD
3-000002B6	CHAR	IO$_READLBLK	3-0000013F	CHAR	IO$_READPBLK
3-00000200	CHAR	IO$_READPRESET	3-00000195	CHAR	IO$_READTRACKD
3-0000033A	CHAR	IO$_READVBLK	3-0000045A	CHAR	IO$_READWTHBUF
3-00000484	CHAR	IO$_READWTHXBUF	3-0000004D	CHAR	IO$_RECAL
3-0000007C	CHAR	IO$_RELEASE	3-000001AB	CHAR	IO$_REREADN
3-000001B8	CHAR	IO$_REREADP	3-000000CA	CHAR	IO$_RETCENTER
3-000002E6	CHAR	IO$_REWIND	3-000002C9	CHAR	IO$_REWINDOFF
3-000000FC	CHAR	IO$_SEARCH	3-00000024	CHAR	IO$_SEEK
3-00000231	CHAR	IO$_SENSECHAR	3-00000309	CHAR	IO$_SENSEMODE
3-0000021D	CHAR	IO$_SETCHAR	3-000003B8	CHAR	IO$_SETCLOCK
3-00000088	CHAR	IO$_SETCLOCKP	3-000002DD	CHAR	IO$_SETMODE
3-000002ED	CHAR	IO$_SKIPFILE	3-000002FA	CHAR	IO$_SKIPRECORD
3-00000029	CHAR	IO$_SPACEFILE	3-0000010E	CHAR	IO$_SPACERECORD
3-000003D7	CHAR	IO$_STARTDATA	3-000000B4	CHAR	IO$_STARTDATAP
3-00000037	CHAR	IO$_STARTMPROC	3-0000020F	CHAR	IO$_STARTSPNDL
3-00000059	CHAR	IO$_STOP	3-0000000D	CHAR	IO$_UNLOAD
3-0000046B	CHAR	IO$_WRITEBUFNCRC	3-0000011E	CHAR	IO$_WRITECHECK
3-000001E4	CHAR	IO$_WRITECHECKH	3-000003FF	CHAR	IO$_WRITECSR
3-00000153	CHAR	IO$_WRITEHEAD	3-000002A2	CHAR	IO$_WRITELBLK
3-00000247	CHAR	IO$_WRITEMARK	3-00000314	CHAR	IO$_WRITEOF
3-0000012A	CHAR	IO$_WRITEPBLK	3-000001C9	CHAR	IO$_WRITERET

```
  3-0000017E  CHAR IO$_WRITETRACKD                3-00000326  CHAR IO$_WRITEVBLK
  3-00000448  CHAR IO$_WRITEWTHBUF                3-00000257  CHAR IO$_WRTTMKR
 AP-00000004a L*1  LUN                            3-000004A1  CHAR QIO_STRING

ARRAYS

    Address  Type  Name                 Bytes  Dimensions
  2-00000000  I*4  QIOCODE                512  (0:1, 0:63)

LABELS

    Address   Label
      **      10

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name              Type  Name
        IRP$W_FUNC         I*4  LIB$EXTZV

COMMAND QUALIFIERS

  FORTRAN /LIS=LIS$:MFTAPE/OBJ=OBJ$:MFTAPE MSRC$:MFTAPE

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19

COMPILATION STATISTICS

  Run Time:                 24.70 seconds
  Elapsed Time:             52.25 seconds
  Page Faults:              580
  Dynamic Memory:           386 pages
```

0151 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MESSAGE.LIS

ML11.LIS

MSCP.LIS

MFTAPE.LIS

MOUNT.LIS

MEMORYS.LIS

MOUVXX.LIS

MCHK_DISP.LIS